#include<stdio.h>

#include<string.h>

#include<stdlib.h>

typedef struct

{

char dirName[20];

char filename[20][20];

int no\_files;

}directory;

typedef struct

{

directory d[10];

int no\_dir;

}MFD;

void single\_level()

{

directory \*dir;

dir=(directory \*)malloc(sizeof(directory));

dir->no\_files=0;

strcpy(dir->dirName,"root");

int ch=0,k=0,i;

while(ch!=3)

{

printf("\n1.Create File\n2.Display Directory\n3.exit\nchoice : ");

scanf("%d",&ch);

if(ch==1)

{

printf("Enter file name : ");

scanf("%s",dir->filename[dir->no\_files]);

for(i=0;i<dir->no\_files;i++)

{

if(!strcmp(dir->filename[i],dir->filename[dir->no\_files]))

{

k=1;

break;

}

}

if(k==1)

{

printf("File name already Exists\n");

}

else

{

printf("File Created\n");

dir->no\_files++;

}

k=0;

}

else if(ch==2)

{

printf("Directory : %s\n",dir->dirName);

printf("File\t\t\tLocation\n");

for(i=0;i<dir->no\_files;i++)

{

printf("%s\t\t\t%s\n",dir->filename[i],dir->dirName);

}

printf("\n");

}

else if(ch!=3)

{

printf("Invalid\n");

}

}

free(dir);

}

void two\_level()

{

MFD \*mdir;

mdir=(MFD \*)malloc(sizeof(MFD));

mdir->no\_dir=0;

directory dir;

dir.no\_files=0;

int ch=0,k=0,i;

int f=0;

while(ch!=5)

{

printf("\n1.Create Directory\n2.Create File\n3.Display Directory\n4.Search file\n5.Exit\nchoice : ");

scanf("%d",&ch);

if(ch==1)

{

k=0;

printf("Enter Directory name : ");

scanf("%s",mdir->d[mdir->no\_dir].dirName);

for(i=0;i<mdir->no\_dir;i++)

{

if(!strcmp(mdir->d[i].dirName,mdir->d[mdir->no\_dir].dirName))

{

k=1;

break;

}

}

if(k==1)

{

printf("Directory already Exists\n");

}

else

{

printf("Directory Created\n");

mdir->d[mdir->no\_dir].no\_files=0;

mdir->no\_dir++;

}

k=0;

}

else if(ch==2)

{

k=-1;

char name[20];

printf("Enter directory : ");

scanf("%s",name);

for(i=0;i<mdir->no\_dir;i++)

{

if(!strcmp(mdir->d[i].dirName,name))

{

k=i;

break;

}

}

if(k!=-1)

{

dir=mdir->d[k];

f=0;

printf("Enter file name : ");

scanf("%s",dir.filename[dir.no\_files]);

for(i=0;i<dir.no\_files;i++)

{

if(!strcmp(dir.filename[i],dir.filename[dir.no\_files]))

{

f=1;

break;

}

}

if(f==1)

{

printf("File name already Exists\n");

}

else

{

printf("File Created\n");

dir.no\_files++;

}

mdir->d[k]=dir;

k=0;

}

else

{

printf("Directory not found\n");

}

}

else if(ch==3)

{

printf("File\t\t\tLocation\n");

for(i=0;i<mdir->no\_dir;i++)

{

for(k=0;k<mdir->d[i].no\_files;k++)

{

printf("%s\t\t\t%s\n",mdir->d[i].filename[k],mdir->d[i].dirName);

}

}

printf("\n");

}

else if(ch==4)

{

char name[20];

printf("Enter file name : ");

scanf("%s",name);

printf("File\t\t\tLocation\n");

for(i=0;i<mdir->no\_dir;i++)

{

for(k=0;k<mdir->d[i].no\_files;k++)

{

if(!strcmp(mdir->d[i].filename[k],name))

printf("%s\t\t\t%s\n",mdir->d[i].filename[k],mdir->d[i].dirName);

}

}

printf("\n");

}

else if(ch!=5)

{

printf("Invalid");

}

}

}

typedef struct

{

char fname[10];

}file;

typedef struct tdir

{

char dname[10];

struct tdir \*dir1,\*dir2,\*dir3;

file \*file1,\*file2;

}tdir;

void treeStruc();

void DAG();

tdir\* init();

void insertDirectory(char pname[],char dirname[]);

void insertFile(char pname[],char filename[]);

void displayContents(tdir \*root, char path[]);

file \*getFilePointer(char path[]);

tdir \*getDirectoryPointer(char path[]);

void createLink(char path[], char dir\_name[]);

void main()

{

int ch=0;

while(ch!=5)

{

printf("\n\t\tFILE ORGANISATION TECHNIQUES\n");

printf("1.Single Level Directory\n2.Two Level Directory\n3.Tree structures directory\n4.DAG\n5.Exit\nChoice :");

scanf("%d",&ch);

switch(ch)

{

case 1 :single\_level();

break;

case 2 :two\_level();

break;

case 3 :treeStruc();

break;

case 4 :DAG();

break;

case 5 :break;

default:printf("\tINVALID !!!\n");

}

}

}

tdir \*t=NULL;

tdir\* init()

{ //initialising root directory

tdir \*root;

root = (tdir \*)malloc(sizeof(tdir));

strcpy(root->dname, "root");

root->dir1 = NULL;

root->dir2 = NULL;

root->dir3 = NULL;

root->file1 = NULL;

root->file2 = NULL;

return root;

}

void insertDirectory(char path[],char dirname[])

{

tdir \*temp = t;

tdir \*tmp;

char \*pname = strtok(path, "/");

pname = strtok(NULL, "/");

while (pname != NULL)

{ //moving to the specified sub-directory

if (temp->dir1 != NULL && strcmp(pname,(tmp=temp->dir1)->dname)==0)

{

temp = temp->dir1;

}

else if (temp->dir2 != NULL && strcmp(pname,(tmp=temp->dir2)->dname)==0)

{

temp = temp->dir2;

}

else if (temp->dir3 != NULL && strcmp(pname,(tmp=temp->dir3)->dname)==0)

{

temp = temp->dir3;

}

pname = strtok(NULL, "/");

}

if (temp->dir1 == NULL || temp->dir2 == NULL || temp->dir3 == NULL)

{ //if space exists in the specified sub-directory

tdir \*new\_dir = (tdir \*)malloc(sizeof(tdir));

new\_dir->dir1 = NULL;

new\_dir->dir2 = NULL;

new\_dir->dir3 = NULL;

new\_dir->file1 = NULL;

new\_dir->file2 = NULL;

strcpy(new\_dir->dname,dirname);

//connecting it to a free pointer of the parent directory

if (temp->dir1 == NULL)

{

temp->dir1 = new\_dir;

printf("Directory created!\n");

}

else if (temp->dir2 == NULL )

{

tmp=temp->dir1;

if(strcmp(dirname,tmp->dname)!=0)

temp->dir2 = new\_dir;

printf("Directory created!\n");

}

else if (strcmp(dirname, (tmp=temp->dir1)->dname) != 0 && strcmp(dirname, (tmp=temp->dir2)->dname) != 0)

{

temp->dir3 = new\_dir;

printf("Directory created!\n");

}

else if (strcmp(dirname, (tmp=temp->dir1)->dname) == 0 || strcmp(dirname, (tmp=temp->dir2)->dname) == 0)

{ //if it already exists

printf("\nDirectory %s already exists!\n", dirname);

}

else

{ //if no space is free in the sub-directory

printf("\nMAX limit reached. Only 3 subdirectories are allowed.\n");

}

}

}

void insertFile(char path[],char filename[])

{

tdir \*temp = t;

tdir \*tmp;

char \*pname = strtok(path, "/");

pname = strtok(NULL, "/");

while (pname != NULL)

{

//moving to specified sub-directory

if (temp->dir1 != NULL && strcmp(pname, (tmp=temp->dir1)->dname) == 0)

{

temp = temp->dir1;

}

else if (temp->dir2 != NULL && strcmp(pname, (tmp=temp->dir2)->dname) == 0)

{

temp = temp->dir2;

}

else if (temp->dir3 != NULL && strcmp(pname, (tmp=temp->dir3)->dname) == 0)

{

temp = temp->dir3;

}

pname = strtok(NULL, "/");

}

if (temp->file1 == NULL || temp->file2 == NULL)

{

//if the sub-directory has space for files

file \*new\_file = (file \*)malloc(sizeof(file));

strcpy(new\_file->fname,filename);

if (temp->file1 == NULL)

{

temp->file1 = new\_file;

}

else if (temp->file2 == NULL)

{

temp->file2 = new\_file;

}

printf("\nFile created!\n");

}

else

{

//if it doesn't have space for files

printf("\nMAX limit reached. Only 2 files are allowed per directory.\n");

}

}

void displayContents(tdir \*root, char path[])

{ //to display the contents of the directory

char temp[50];

if (root != NULL)

{

strcat(path,root->dname);

strcat(path, "/");

if (root->file1 != NULL)

{

printf("%s\t\t\t\t%s\n", root->file1->fname, path);

}

if (root->file2 != NULL)

{

printf("%s\t\t\t\t%s\n", root->file2->fname, path);

}

if (root->dir1 != NULL)

{

strcpy(temp,path);

displayContents(root->dir1,path);

}

if (root->dir2 != NULL)

{

strcpy(temp,path);

displayContents(root->dir2, temp);

}

if (root->dir3 != NULL)

{

strcpy(temp, path);

displayContents(root->dir3, temp);

}

}

}

void treeStruc()

{

int choice;

char name[20];

char path[20];

t=init();

do

{

printf("\n1.Create Directory\n");

printf("2.Create file\n");

printf("3.Display file\n");

printf("4.Exit\n");

printf("Enter choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\nEnter the name of the directory:");

scanf("%s",&name);

printf("Enter the path where you want to store the directory:");

scanf("%s",&path);

insertDirectory(path,name);

break;

case 2:

printf("\nEnter the name of the file:");

scanf("%s",&name);

printf("Enter the path where you want to store the file:");

scanf("%s",&path);

insertFile(path,name);

break;

case 3:

strcpy(path, "");

printf("\nFile\t\t\t\tPath\n\n");

displayContents(t,path);

break;

case 4:

break;

}

}while(choice!=4);

}

file \*getFilePointer(char path[])

{ //to return the file pointer to the specified file

tdir \*temp = t;

tdir \*tmp;

file \*f;

char \*split = strtok(path, "/");

char \*fname;

while (split != NULL)

{ //traversing to the specified sub-directory

if (temp->dir1 != NULL && strcmp(split,(tmp=temp->dir1)->dname) == 0)

{

temp = temp->dir1;

}

else if (temp->dir2 != NULL && strcmp(split,(tmp=temp->dir2)->dname) == 0)

{

temp = temp->dir2;

}

else if (temp->dir3 != NULL && strcmp(split, (tmp=temp->dir3)->dname) == 0)

{

temp = temp->dir3;

}

fname = split;

split = strtok(NULL, "/");

if (split == NULL)

{ //reached the parent directory of the file

if (strcmp((f=temp->file1)->fname, fname) == 0)

{

return temp->file1;

}

else if (strcmp((f=temp->file2)->fname,fname) == 0)

{

return temp->file2;

}

else

{

printf("\nThe specified file does not exist.\n");

return NULL;

}

}

}

return NULL;

}

tdir \*getDirectoryPointer(char path[])

{ //to return the directory pointer to the specified directory

char \*split = strtok(path, "/");

tdir \*temp = t;

tdir \*tmp;

while (split != NULL)

{ //traversing to the specified sub-directory

if (temp->dir1 != NULL && strcmp(split,(tmp=temp->dir1)->dname) == 0)

{

temp = temp->dir1;

}

else if (temp->dir2 != NULL && strcmp(split,(tmp=temp->dir2)->dname) == 0)

{

temp = temp->dir2;

}

else if (temp->dir3 != NULL && strcmp(split, (tmp=temp->dir3)->dname) == 0)

{

temp = temp->dir3;

}

split = strtok(NULL, "/");

if (split == NULL)

{ //reached the required directory

return temp;

}

}

return NULL;

}

void createLink(char path[], char dir\_name[])

{ //creating a link to existing file to another directory

file \*temp\_file = getFilePointer(path);

tdir \*temp\_dir = getDirectoryPointer(dir\_name);

if (temp\_file != NULL)

{

if (temp\_dir->file1 == NULL)

{

temp\_dir->file1 = temp\_file;

}

else if (temp\_dir->file2 == NULL)

{

temp\_dir->file2 = temp\_file;

}

else

{

printf("\nLink cannot be created.\n");

}

}

}

void DAG()

{

int choice;

char name[20];

char path[20];

t=init();

do

{

printf("\n1.Create Directory\n");

printf("2.Create file\n");

printf("3.Create a link to a file\n");

printf("4.Display file\n");

printf("5.Exit\n");

printf("Enter choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\nEnter the name of the directory:");

scanf("%s",&name);

printf("Enter the path where you want to store the directory:");

scanf("%s",&path);

insertDirectory(path,name);

break;

case 2:

printf("\nEnter the name of the file:");

scanf("%s",&name);

printf("Enter the path where you want to store the file:");

scanf("%s",&path);

insertFile(path,name);

break;

case 3:

printf("\nEnter the Path of File (Including File Name): ");

scanf("%s", path);

printf("\nEnter the Path of Directory to Create Link in: ");

scanf("%s",name);

createLink(path,name);

break;

case 4:

strcpy(path, "");

printf("\nFile\t\t\t\tPath\n\n");

displayContents(t,path);

break;

case 5:

break;

}

}while(choice!=5);

}

/\*

PS F:\SEM4\OS\Assignment12> gcc -o k fileorg.c PS F:\SEM4\OS\Assignment12> ./k

FILE ORGANISATION TECHNIQUES

1.Single Level Directory

2.Two Level Directory

3.Tree structures directory

4.DAG

5.Exit

Choice :1

1.Create File

2.Display Directory

3.exit

choice : 1

Enter file name : 1

File Created

1.Create File

2.Display Directory

3.exit

choice : 1

Enter file name : 2

File Created

1.Create File

2.Display Directory

3.exit

choice : 1

Enter file name : 3

File Created

1.Create File

2.Display Directory

3.exit

choice : 2

Directory : root

File Location

1 root

2 root

3 root

1.Create File

2.Display Directory

3.exit

choice : 3

FILE ORGANISATION TECHNIQUES

1.Single Level Directory

2.Two Level Directory

3.Tree structures directory

4.DAG

5.Exit

Choice :2

1.Create Directory

2.Create File

3.Display Directory

4.Search file

5.Exit

choice : 1

Enter Directory name : a

Directory Created

1.Create Directory

2.Create File

3.Display Directory

4.Search file

5.Exit

choice : 1

Enter Directory name : b

Directory Created

1.Create Directory

2.Create File

3.Display Directory

4.Search file

5.Exit

choice : 1

Enter Directory name : c

Directory Created

1.Create Directory

2.Create File

3.Display Directory

4.Search file

5.Exit

choice : 2

Enter directory : a

Enter file name : 1

File Created

1.Create Directory

2.Create File

3.Display Directory

4.Search file

5.Exit

choice : 2

Enter directory : a

Enter file name : 2

File Created

1.Create Directory

2.Create File

3.Display Directory

4.Search file

5.Exit

choice : 2

Enter directory : a

Enter file name : 3

File Created

1.Create Directory

2.Create File

3.Display Directory

4.Search file

5.Exit

choice : 2

Enter directory : b

Enter file name : 2

File Created

1.Create Directory

2.Create File

3.Display Directory

4.Search file

5.Exit

choice : 2

Enter directory : c

Enter file name : 1

File Created

1.Create Directory

2.Create File

3.Display Directory

4.Search file

5.Exit

choice : 3

File Location

1 a

2 a

3 a

2 b

1 c

1.Create Directory

2.Create File

3.Display Directory

4.Search file

5.Exit

choice : 4

Enter file name : 1

File Location

1 a

1 c

1.Create Directory

2.Create File

3.Display Directory

4.Search file

5.Exit

choice : 5

FILE ORGANISATION TECHNIQUES

1.Single Level Directory

2.Two Level Directory

3.Tree structures directory

4.DAG

5.Exit

Choice :3

1.Create Directory

2.Create file

3.Display file

4.Exit

Enter choice:1

Enter the name of the directory:a

Enter the path where you want to store the directory:root

Directory created!

1.Create Directory

2.Create file

3.Display file

4.Exit

Enter choice:1

Enter the name of the directory:b

Enter the path where you want to store the directory:root/a

Directory created!

1.Create Directory

2.Create file

3.Display file

4.Exit

Enter choice:2

Enter the name of the file:1

Enter the path where you want to store the file:root

File created!

1.Create Directory

2.Create file

3.Display file

4.Exit

Enter choice:2

Enter the name of the file:2

Enter the path where you want to store the file:root/a

File created!

1.Create Directory

2.Create file

3.Display file

4.Exit

Enter choice:3

File Path

1 root/

2 root/a/

1.Create Directory

2.Create file

3.Display file

4.Exit

Enter choice:4

FILE ORGANISATION TECHNIQUES

1.Single Level Directory

2.Two Level Directory

3.Tree structures directory

4.DAG

5.Exit

Choice :4

1.Create Directory

2.Create file

3.Create a link to a file

4.Display file

5.Exit

Enter choice:1

Enter the name of the directory:a

Enter the path where you want to store the directory:root

Directory created!

1.Create Directory

2.Create file

3.Create a link to a file

4.Display file

5.Exit

Enter choice:1

Enter the name of the directory:b

Enter the path where you want to store the directory:root/a

Directory created!

1.Create Directory

2.Create file

3.Create a link to a file

4.Display file

5.Exit

Enter choice:2

Enter the name of the file:1

Enter the path where you want to store the file:root/a

File created!

1.Create Directory

2.Create file

3.Create a link to a file

4.Display file

5.Exit

Enter choice:2

Enter the name of the file:2

Enter the path where you want to store the file:root/a/b

File created!

1.Create Directory

2.Create file

3.Create a link to a file

4.Display file

5.Exit

Enter choice:3

Enter the Path of File (Including File Name): root/a/1

Enter the Path of Directory to Create Link in: root/a/b

1.Create Directory

2.Create file

3.Create a link to a file

4.Display file

5.Exit

Enter choice:4

File Path

1 root/a/

2 root/a/b/

1 root/a/b/

1.Create Directory

2.Create file

3.Create a link to a file

4.Display file

5.Exit

Enter choice:5

FILE ORGANISATION TECHNIQUES

1.Single Level Directory

2.Two Level Directory

3.Tree structures directory

4.DAG

5.Exit

Choice :5

PS F:\SEM4\OS\Assignment12>

\*/